CPE403 – Advanced Embedded Systems

Design Assignment 1

Name: Jenifer Christina

Email: chrisj14@unlv.nevada.edu

Github Repository link: https://github.com/chrisj14/CCS-Assignment

Youtube Playlist link: https://youtu.be/cwDEEj2b8To

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

Task 02: Continue with Task 01, develop an user interface using UART to perform the following: Enter the cmd:
R: Red LED,
G: Green LED,
B: Blue LED,
T: Temperature,
S: status of the LEDs.
Based on the command (cmd) the program should turn ON Red LED when R is entered in the terminal, etc. Command of ‘r’ will turn off the Red LED.’T’ reads Temperature in Centigrade, and ‘t’ read Temperature in Fahrenheit. ‘S’ read status of the RGB LEDs.

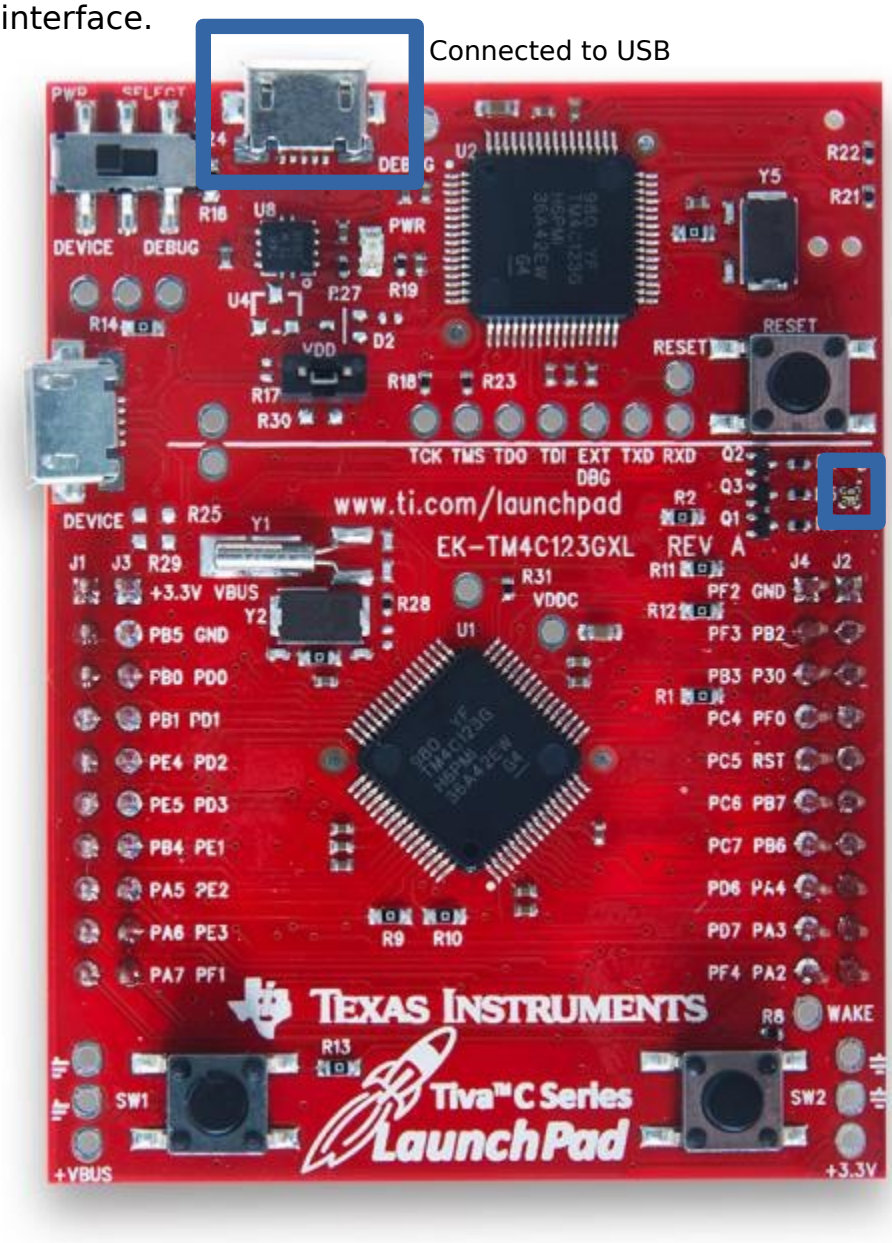
```
/* ----------------------          Include Files       ---------------------
*/
#include <stdint.h>             // Library of Standard Integer Types
#include <stdbool.h>            // Library of Standard Boolean Types
#include "inc/tm4c123gh6pm.h"   //def. for the interrupt and register
assignments on the Tiva C Series device on the launchPad board
#include "inc/hw_memmap.h"      // Macros defining the memory map of the Tiva
C Series device
#include "inc/hw_gpio.h"        // Defines macros for GPIO hardware
#include "inc/hw_types.h"       // Defines common types and macros
#include "driverlib/sysctl.h"   // Defines and macros for System Control API
of DriverLib
#include "driverlib/pin_map.h"  // Mapping of peripherals to pins of all
parts
```

```c
#include "driverlib/gpio.h"     // Defines and macros for GPIO API of
DriverLib
#include "driverlib/interrupt.h"//defines & macros for NVIC
Controller(Interrupt)API of driverlib.
#include "driverlib/timer.h"    //Defines and macros for Timer API of
driverLib.
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/debug.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include <string.h>

#ifdef DEBUG
void_error_(vhar *pcFilename, uint32_t ui32Line){}
#endif

/* ----------------------      Global Variables        ---------------------
*/
uint32_t ui32Period;
char buffer [4];
uint32_t ui32ADC0Value[4];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

/* ----------------------     Function Prototypes      ---------------------
*/
void UARTIntHandler(void);

/* ----------------------          Main Program        ---------------------
*/
int main(void) {
    //System clock to 40Mhz (PLL= 400Mhz / 10 = 40Mhz)
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);
    //Configure peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    //Setup for ADC
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|
ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);
    //Setup for UART
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
```

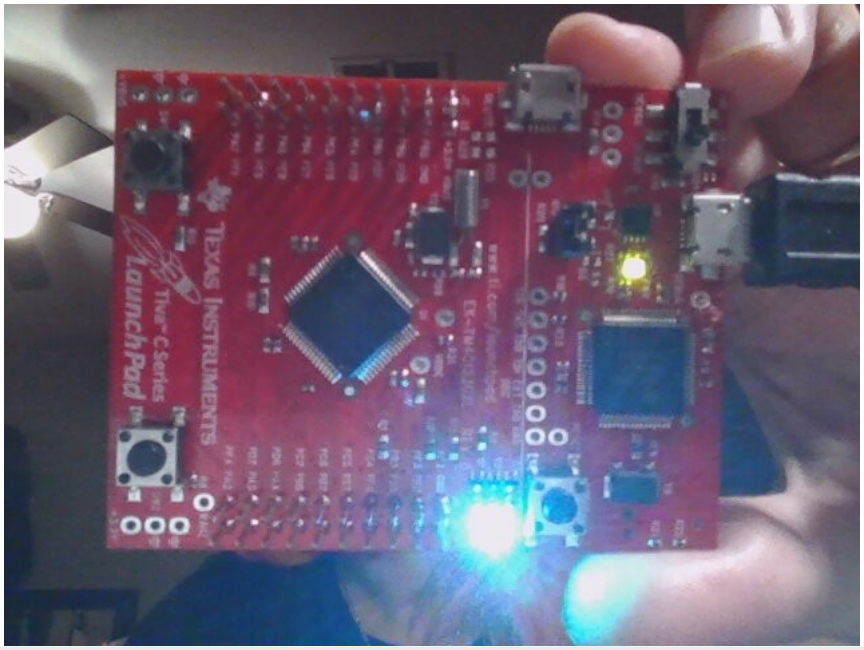
```
    UARTStdioConfig(0, 115200, 16000000);

    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
//enable pin for LED PF2

    IntMasterEnable(); //enable processor interrupts
    IntEnable(INT_UART0); //enable the UART interrupt
    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT); //only enable RX
and TX interrupts

    UARTprintf("Enter the cmd: \n"
               "R: Red LED, \n"
               "G: Green LED, \n"
               "B: Blue LED, \n"
               "T: Temperature, \n"
               "S: status of the LEDs. \n");

    while (1){}

}

void UARTIntHandler(void)
{
    uint32_t ui32Status;

    ui32Status = UARTIntStatus(UART0_BASE, true); //get interrupt status

    UARTIntClear(UART0_BASE, ui32Status); //clear the asserted interrupts

    while(UARTCharsAvail(UART0_BASE)) //loop while there are chars
    {
        char cChar=UARTCharGet(UART0_BASE);
        UARTCharPutNonBlocking(UART0_BASE, cChar); //echo character
        if (cChar=='R') {       //Turn on RED LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);
        }
        else if (cChar=='r') {  //Turn off RED LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
        }
        else if (cChar=='G') {  //Turn on Green LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, GPIO_PIN_3);
        }
        else if (cChar=='g') {  //Turn off Green LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0);
        }
        else if (cChar=='B') {  //Turn on Blue LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
        }
        else if (cChar=='b') {  //Turn off Blue LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
        }
        else if (cChar=='T') {  //Show Temperature in Centigrade
            ADCIntClear(ADC0_BASE,2);
            ADCProcessorTrigger(ADC0_BASE, 2);
```

```
            ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);

            ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] +
ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
            ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
            ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
            UARTprintf("\n C %3d\t \n",ui32TempValueC );
        }
        else if (cChar=='t') {  //Show Temperature in Farenheit
            ADCIntClear(ADC0_BASE,2);
            ADCProcessorTrigger(ADC0_BASE, 2);

            ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);

            ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] +
ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
            ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
            ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
            UARTprintf("\n F %3d\t \n",ui32TempValueF );
        }
        else if (cChar=='S') {  //Show LED Status
            UARTprintf("\n");
            if (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_1))
                UARTprintf("Red LED is on \n");
            if (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
                UARTprintf("Blue LED is on \n");
            if (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_3))
                UARTprintf("Green LED is on \n");
        }
    }


}
```

2. Block diagram and/or Schematics showing the components, pins used, and interface.



3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.



Terminal

/dev/ttyACM0

```
Enter the cmd:
R: Red LED,
G: Green LED,
B: Blue LED,
T: Temperature,
S: status of the LEDs.
T
 C 147
T
 C  23
T
 C  23
t
 F  71
t
 F  73
RrGBS
Blue LED is on
Green LED is on
gbS
RS
Red LED is on
r
```

4. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

“This assignment submission is my own, original work”.
Jenifer Christina